Manuel Strasser

ANGABE

Die Sicherheitsfahrschaltung

Im Zugsverkehr ist die Sicherheit der Fahrgäste oberstes Gebot. Die Gesundheit des Triebfahrzeugführers muss, daher überwacht werden, um im Notfall entsprechende Maßnahmen einzuleiten. Diese Überwachung erfolgt mittels einem Taster, der alle 30 Sekunden kurz unterbrochen werden muss.

- Hält der Triebfahrzeugführer die Taste länger als 30 Sekunden lang gedrückt, erfolgt ein Warnsignal.
- Nach weiteren 15 Sekunden wird der Zug notgebremst.
- Ignorieren Sie den Fall, wenn der Taster nicht gedrückt wird.

Entwickeln Sie die Sicherheitsfahrschaltung mithilfe eines Atmega328p.

1. Wählen Sie geeignete Ein- und Ausgänge für:
   - den Taster,
   - das Warnsignal (in diesem Fall eine LED) und
   - das digitale Ausgangssignal für die Notbremse (logisch 1 => Notbremse)
2. Zeitmessung:
   - Konfigurieren Sie einen beliebigen Timer, der jede Sekunde einen Interrupt auslöst
   - Zählen Sie die Sekunden und setzen Sie die Ausgänge entsprechend der abgelaufenen Zeit
3. Taster:
   - Konfigurieren Sie Ihre verwendeten Ein- und Ausgänge
   - Verwenden Sie Interrupts, um die Tasterstellung einzulesen.
4. Schaltplan:
   - Zeichnen Sie einen Schaltplan ihres Systems!

Abgabe

- Drucken Sie Ihren Sourcecode aus (Name!)
- Geben Sie den Schaltplan ab.

Punkte

- Teilaufgabe 1: 10 Punkte 10P
- Teilaufgabe 2: 10 Punkte 10P
- Teilaufgabe 3: 10 Punkte 10P
- Teilaufgabe 4: 10 Punkte 9P

Gesamt 40 Punkte 39P

- Sehr Gut: 36,2 - 40,0
- Gut: 32,2 - 36,0
- Befriedigend: 26,2 - 32,0
- Genügend: 20,2 - 26,0
- Nicht Genügend: 0,0 - 20,0

- Timer 1s Intervall

16bit Timer verwendet $f = 1s$ $T = 16MHz$

$$P = \frac{16 \cdot 10^6 \cdot 1}{2^{16}} = 244{,}14 \approx 256 \text{ Prescaler benötigt}$$

$$\text{Schritte} = \frac{16 \cdot 10^6 \cdot 1}{256} = 62\,500 \text{ Schritte für 1s}$$

- Schalter - Schaltung

PORTC1 → PCINT9

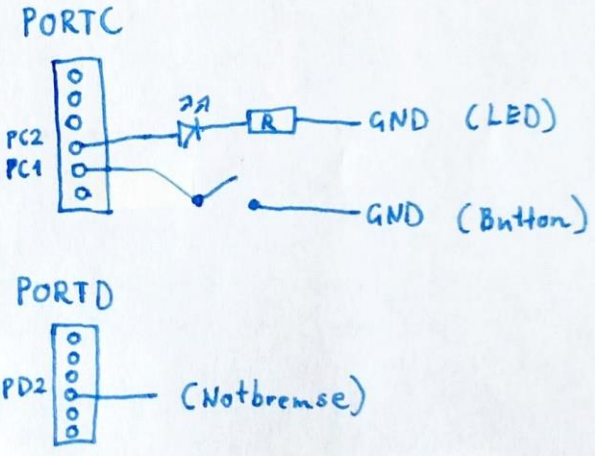
Pull-Up Intern verwendet

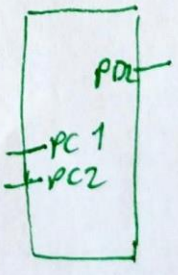
- LED - Schaltung

- Notbremse

PORTD2

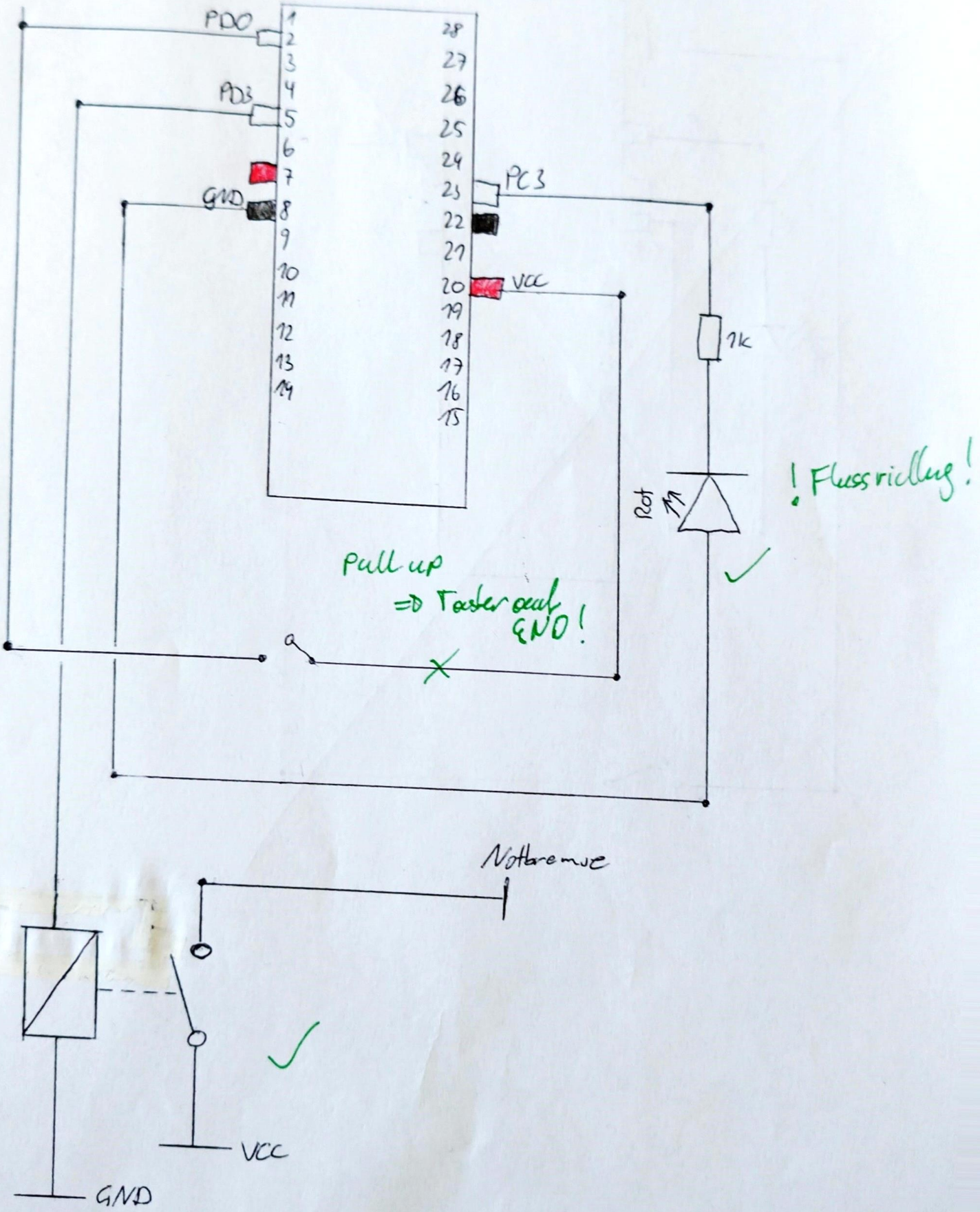
Ino Ardu :)





Verwenden!

Johannes Allinger



```
/*
 * SYTI_Test.c
 *
 * Created: 23.02.2023 10:44:53
 * Author : Manuel Strasser | 5CHIT
 */

#define F_CPU 16000000
#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <util/atomic.h>
#include <util/delay.h>

//Zählher des Timers
volatile int timercnt=0;

//Button-Status -> 0: nicht gedrückt, 1: gedrückt
volatile int buttonStatus=0;

//Status, ob der Timer für das Warnsignal läuft -> 0: false, 1: true
volatile int iscountingForSignal = 0;

//Status, ob der Timer für die Notbremse läuft -> 0: false, 1: true
volatile int iscountingForStop = 0;

void init();
void init_timer();
void TurnOnLED();
void StopTrain();

int main(void)
{
    init();
    init_timer();

    //Interrupts aktivieren
    sei();

    while (1)
    {
        //Atomic Block, damit Interrupts während des Timer Counter au
unterdrückt werden
        ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
        {
            if (iscountingForSignal == 1 && timercnt >= 30) //überprü
Button 30s lang gedrückt wurde
            {
                iscountingForSignal = 0;
                iscountingForStop = 1;
                timercnt = 0;
                TurnOnLED();
            }

            if (iscountingForStop == 1 && timercnt >= 15) //überprüfe
```

```
            }

            if (buttonStatus == 1 && iscountingForSignal == 0) //wenn
 gedrⁿckt wird, das 30s Intervall starten
            {
                //Timer zurⁿcksetzen, um ab jetzt 30 Sekunden zEhlen
                timercnt = 0;
                iscountingForSignal = 1;
            }
            else if (buttonStatus == 0 && iscountingForSignal == 1) /
 Button nicht mehr gedrⁿckt wird, 30s Timer ignorieren
            {
                iscountingForSignal = 0;
            }
        }

    }
}

//Register fⁿr LED, Button und Notbremse konfigurieren
void init()
{
    //LED Port -> PORTC2 auf Output -> HIGH: LED an, LOW: LED aus
    DDRC |= (1<<PORTC2);

    //Notbremse Port -> PORTD2 auf Output -> HIGH: Notbremse an, LOW:
    DDRD |= (1<<PORTD2);

    //Button Interrupt Register setzen
    //PORT auf HIGH, fⁿr internen Pull Up
    PORTC |= (1<<PORTC1);
    //Pin Change Mask Register setzen
    PCMSK1 |= (1<<PCINT9);
    //Pin Change Interrupt aktivieren
    PCICR |= (1<<PCIE1);

}

//Register fⁿr 16bit Timer konfigurieren
void init_timer()
{
    // Timer CTC Modus aktivieren
    TCCR1B |= (1<<WGM12);
    // Prescaler auf 256
    TCCR1B |= (1<<CS12);
    // Output Compare A Interrupt aktivieren
    TIMSK1 |= (1<<OCIE1A);
    // Compare Wert setzen -> F_CPU *  1s / 256 (Prescaler) = 62500 S
Intervall zu erreichen
    OCR1A = 62500;
}

//wird aufgerufen, wenn Warnsignal erfolgt
void TurnOnLED()
{
```

Manuel Strasser

```
weitere 15s vergangen sind
                {
                    iscountingForStop = 0;
                    StopTrain();
```

Manuel Strasser

```
        //LED Port auf HIGH
        PORTC |= (1<<PORTC2);
}
```

```
//wird aufgerufen, wenn Notbremse erfolgt
void StopTrain()
{
    //Port fnr Notbremse auf HIGH
    PORTD |= (1<<PORTD2);
}

//Interrupt Routine fnr Button Interrupt
ISR(PCINT1_vect)
{
    //wenn Button gedrnckt
    if(!(PINC & (1<<PINC1)))
    {
        buttonStatus = 1;
    }
    else
    {
        buttonStatus = 0;
    }
}

//Interrupt Routine fnr Timer Compare Interrupt
ISR(TIMER1_COMPA_vect)
{
    timercnt++; //Timer Counter wird somit alle 1s um 1 erhöht
}
```